Scoreboarding

In a dynamically scheduled pipeline, all instructions pass through the issue stage in order—this is known as *in-order issue*; however, they can be stalled or bypass each other in the second stage, called the *read operands* stage, and thus enter execution *out of order*.

Scoreboarding is a technique for allowing instructions to execute out of order when there are sufficient resources and no data dependences.

Scoreboarding takes its name from the CDC 6600's scoreboard, which developed this capability.

The goal of a scoreboard is to maintain an execution rate of one instruction per clock cycle (when there are no structural hazards) by executing an instruction as early as possible. Thus, when the next instruction to execute is stalled, other instructions can be issued and executed if they do not depend on any active or stalled instruction.

The scoreboard takes full responsibility for instruction issue and execution, including all hazard detection. Taking advantage of out-of-order execution requires multiple instructions to be in their EX stage simultaneously. This can be achieved with multiple functional units, pipelined functional units, or both.

Every instruction goes through the scoreboard, where a record of the data dependences is constructed. This step corresponds to the instruction issue and replaces part of the ID step in the MIPS pipeline.

The scoreboard then determines when the instruction can read its operands and begin execution. If the scoreboard decides the instruction cannot execute immediately, it monitors every change in the hardware and decides when the instruction can execute.

The scoreboard also controls when an instruction can write its result into the destination register. Thus, all hazard detection and resolution are centralised in the scoreboard.

When using scoreboarding, each instruction undergoes four steps in executing. These steps replace the ID, EX and WB steps in the standard MIPS pipeline and are:

Step	Description
Issue	If a functional unit for the instruction is free and no other active instruction has the same destination register, the scoreboard issues the instruction to the functional unit and updates its internal data structure. This replaces a portion of the ID step in the MIPS pipeline. By ensuring that no other active functional unit wants to write its result into the destination register, we guarantee that write-after-write (WAW) hazards cannot be present.
Read operands	The scoreboard monitors the availability of the source operands. A source operand is available if no earlier issued active instruction is going to write it. When the source operands are available, the scoreboard tells the functional unit to proceed to read the operands from the registers and begin execution. The scoreboard resolves read-after-write (RAW) hazards dynamically in this step, and instructions may be sent into execution out of order. This step completes the function of the ID step in the MIPS pipeline.
Execution	The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard that it has completed. This step replaces the EX step in the MIPS pipeline.
Write result	Once the scoreboard is aware that the functional unit has completed execution, the scoreboard checks for write-after-read (WAR) hazards and stalls the completing instruction if necessary. A completing instruction cannot be allowed to write its results when there is an instruction that has not read its operands that precedes it or if one of the operands is the same register as the result of the completing instruction. If the WAR hazard does not exist, or when it clears, the scoreboard tells the functional unit to store its result to the destination register. This step replaces the WB step in the MIPS pipeline.

Because the operands for an instruction are read only when both operands are available in the register file, this scoreboard does not take advantage of forwarding. instead, registers are only read when they are both available.